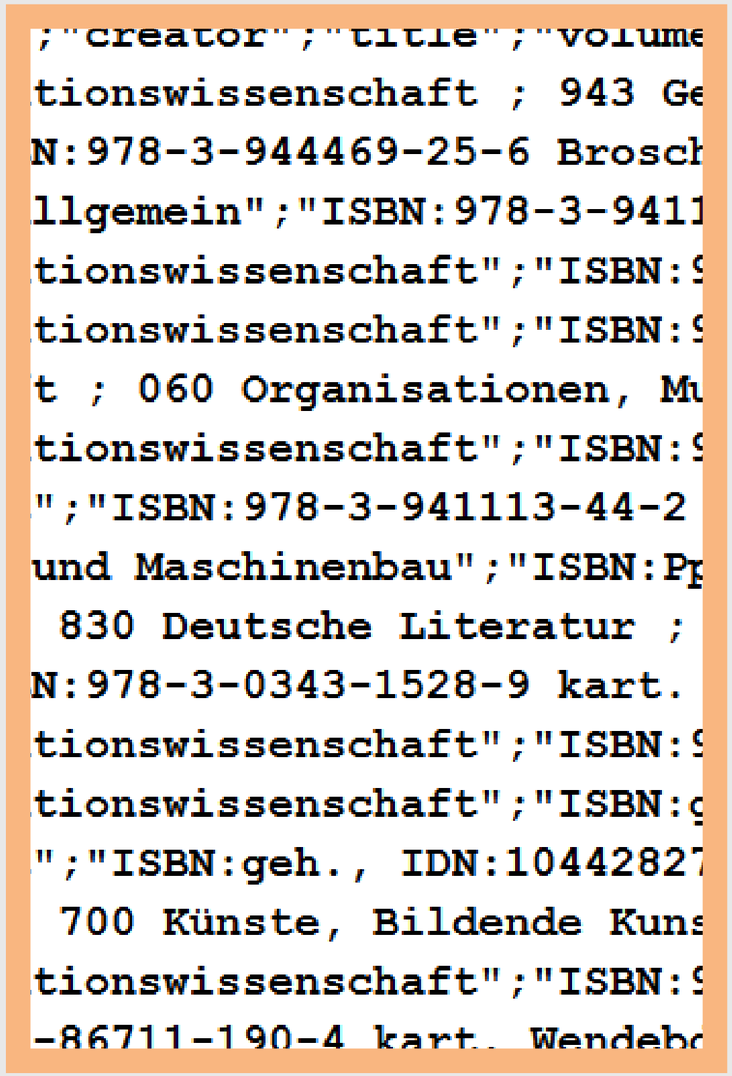
APACHE COMMONS CSV

DOMINIKA BUJNAROWSKA
JAKUB CHACHOWSKI

TABLE OF CONTENT



APACHE COMMONS CSV

Apache Commons CSV to biblioteka Java z Apache Commons, która umożliwia łatwe przetwarzanie plików CSV (Comma-Separated Values).
Biblioteka oferuje zestaw narzędzi do odczytu i zapisu plików CSV w prosty sposób, eliminując potrzebę samodzielnego parsowania danych.
Obsługuje różne formaty CSV (np. RFC4180, MySQL, Oracle), umożliwia konfigurację delimitera, cudzysłowów i innych ustawień formatu. Dodatkowo pozwala na pracę z nagłówkami kolumn i ułatwia iterację po rekordach w czytelnej formie.

DODANIE BIBLIOTEKI DO PROJEKTU MAVEN

Przed instalacją warto sprawdzić aktualną wersję na stronie:
https://commons.apache.org/proper/commons-csv/dependency-info.html
Dodaj zależność w pliku pom.xml, a następnie zaktualizuj zależności uruchamiając komendę
mvn clean install

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-csv</artifactId>
    <version>1.12.0</version>
</dependency>
```

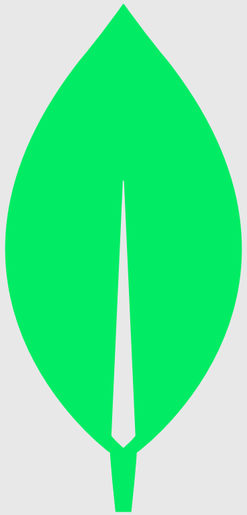
GŁÓWNE FUNKCJONALNOŚCI

- odczyt i zapis plików CSV
- przechodzenie po wierszach/kolumnach
- obsługa nagłówków
- zarządzanie separatorami i cudzysłowami
- obsługa błędów

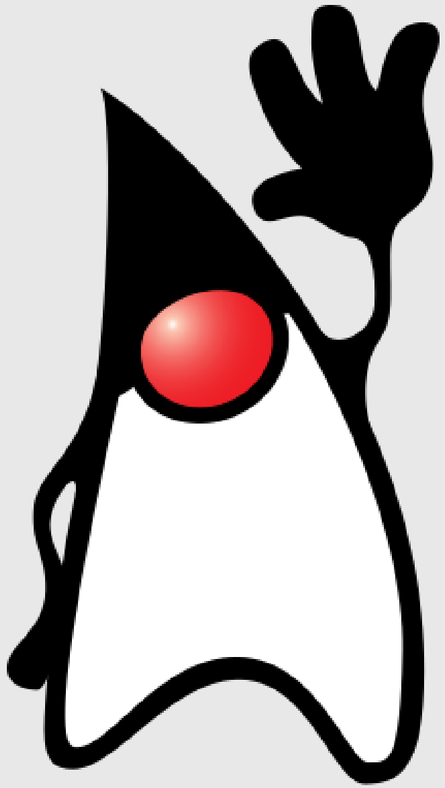
dOsoby	Imie	Nazwisko	NrRejestracyjny
1	Echo	Ayala	FVX4190
2	Nolan	Stein	DUG5882
3	Lee	Joseph	TBG6984
4	Clayton	Powers	AVK5773
5	Lance	Hudson	PHT3279
6	Jada	Bruce	DFA4984
7	Lucy	Eaton	IVO5023
8	Brynne	Franco	QUF2154
9	Nola	Simpson	OCG9925
10	Alexa	Payne	TJN4918
11	Selma	Dickson	KMK1804
12	Xyla	Norman	SJX2705
13	Jordan	Roberts	SVD7310
14	Zeus	Butler	MWU6374
15	Lucius	Adkins	AJA1058
16	Zorita	Tate	OII3224
17	Alice	Payne	VFM2557
18	Quemby	Melendez	NWB8596
19	Chiquita	Wilkerson	QPB2071
20	Giacomo	Richards	IZZ5265
21	Illana	Ryan	YVG2064

OBSŁUGIWANE FORMATY CSV

- DEFAULT (RFC 4180)
- EXCEL
- INFORMIX_UNLOAD
- INFORMIX_UNLOAD_CSV
- MONGODB_CSV
- MONGODB_TSV
- MYSQL
- ORACLE
- POSTGRESQL_CSV
- POSTGRESQL_TEXT
- RFC4180



PRZYKŁADY UŻYCIA



Przykładowy kod zapisujący do pliku

```java
package CSVFormatDefault;

import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVPrinter;

import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;

public class CSVWriter {
    private static final String CSV_FILE_PATH = "./sample.csv";  1 usage

    public static void main(String[] args) {
        try (Writer out = new FileWriter(CSV_FILE_PATH);
             CSVPrinter csvPrinter = new CSVPrinter(out, CSVFormat.DEFAULT.withHeader("Name", "Age", "City"))) {

            // Zapisanie rekordu
            csvPrinter.printRecord( ...values: "Anna","28","Warsaw");
            csvPrinter.printRecord( ...values: "Krzysztof","35","Krakow");
            csvPrinter.printRecord( ...values: "Maria","22","Wroclaw");
            csvPrinter.printRecord( ...values: "Tomasz","41","Gdansk");

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Przykładowy kod odczytujący dane z pliku

```java
public class CSVReader {
    private static final String CSV_FILE_PATH = "./sample.csv";  2 usages

    public static void main(String[] args) {
        ReadAll();
        Filtered();
    }

    public static void ReadAll() {  1 usage
        try (
                Reader reader = new FileReader(CSV_FILE_PATH);
                CSVParser csvParser = new CSVParser(reader, CSVFormat.DEFAULT
                        .withFirstRecordAsHeader()
                        .withIgnoreHeaderCase()
                        .withTrim())
        ) {
            for (CSVRecord csvRecord : csvParser) {
                String name = csvRecord.get("name");  // po nazwie kolumny
                String age = csvRecord.get(1);        // po indeksie kolumny
                String city = csvRecord.get(2);

                System.out.println("Record No - " + csvRecord.getRecordNumber());
                System.out.println("---------------");
                System.out.println("Name : " + name);
                System.out.println("Age : " + age);
                System.out.println("City : " + city);
                System.out.println("---------------\n\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Przykładowy kod filtrujący dane

```java
for (CSVRecord csvRecord : csvParser) {
    String name = csvRecord.get("Name");
    String ageString = csvRecord.get("Age");
    String city = csvRecord.get("City");

    try {
        int age = Integer.parseInt(ageString);

        if (age > 30) {
            count++; // Zliczanie rekordów
            System.out.println("Record No - " + csvRecord.getRecordNumber());
            System.out.println("---------------");
            System.out.println("Name : " + name);
            System.out.println("Age : " + age);
            System.out.println("City : " + city);
            System.out.println("---------------\n\n");
        }
```

Przykładowy kod mapujący rekordy na obiekty

```java
package CSVFormatDefault;

public class Person {  4 usages
    private String name;  2 usages
    private String age;  2 usages
    private String city;  2 usages

    // Constructor
    public Person(String name, String age, String city) {
        this.name = name;
        this.age = age;
        this.city = city;
    }
```

```java
public static void main(String[] args) {
    List<Person> people = new ArrayList<>();

    try (Reader reader = new FileReader(CSV_FILE_PATH);
         CSVParser csvParser = new CSVParser(reader, CSVFormat.DEFAULT
                 .withFirstRecordAsHeader()
                 .withIgnoreHeaderCase()
                 .withTrim())) {

        for (CSVRecord csvRecord : csvParser) {
            String name = csvRecord.get("Name");
            String age = csvRecord.get("Age");
            String city = csvRecord.get("City");

            Person person = new Person(name, age, city);
            people.add(person);
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
```

Przykładowy kod odczytujący dane z pliku .csv wygenerowanego przez MS Excel

Uwaga! Pilki wygenerowane przez Excel'a mają różne separatory w zależności od ustawień regionalnych

```
public class CSVReader {
    private static final String CSV_FILE_PATH =  1 usage
            "ścieżka\\do\\pliku\\excel_exported.csv";

    public static void main(String[] args) {
        try (Reader reader = new FileReader(CSV_FILE_PATH);
             CSVParser csvParser = new CSVParser(reader, CSVFormat.EXCEL
                     .withDelimiter(';')                    // !
                     .withFirstRecordAsHeader())) {

            for (CSVRecord record : csvParser) {
                String IdOsoby = record.get(0);
                String Imie = record.get(1);
                String Nazwisko = record.get(2);
                String NrRejestracyjny = record.get(3);
                System.out.println("IdOsoby: " + IdOsoby + ", Imie: "
                        + Imie + ", Nazwisko: " + Nazwisko +
                        ", NrRejestracyjny: " + NrRejestracyjny);
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Przykładowy kod dodający kolumnę danych do pliku

```java
public class CSVAddColumn {
    public static void main(String[] args) {
        String[] countries = {"Poland", "Poland", "Czech Republic", "Poland"};

        try (Reader reader = new FileReader( fileName: "sample.csv");
             CSVParser csvParser = new CSVParser(reader, CSVFormat.DEFAULT
                     .withFirstRecordAsHeader());
             Writer writer = new FileWriter( fileName: "output.csv");
             CSVPrinter csvPrinter = new CSVPrinter(writer, CSVFormat.DEFAULT
                     .withHeader("Name", "Age", "City", "Country"))) {

            int rowIndex = 0;
            for (CSVRecord record : csvParser) {
                String name = record.get("Name");
                String age = record.get("Age");
                String city = record.get("City");

                String country = countries[rowIndex];

                csvPrinter.printRecord(name, age, city, country);
                rowIndex++;
            }

            csvPrinter.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

WAŻNE FUNKCJE

- pomijanie pustych wierszy

```
CSVParser csvParser = new CSVParser(reader, CSVFormat.DEFAULT.withHeader().withIgnoreEmptyLines());
```

- wartości domyślne dla brakujących danych

```
for (CSVRecord csvRecord : csvParser) {
        String name = csvRecord.isSet("Name") ? csvRecord.get("Name") : "Unknown";
        String position = csvRecord.isSet("Position") ? csvRecord.get("Position") : "Employee";
        String company = csvRecord.isSet("Company") ? csvRecord.get("Company") : "Unknown Company";

        Person person = new Person(name, position, company);
        people.add(person);
}
```

- obsługa pustych wartości jako null

```
CSVParser csvParser = new CSVParser(reader, CSVFormat.DEFAULT.withHeader().withNullString(""));
```

- pomijanie brakujących kolumn

```
CSVParser csvParser = new CSVParser(reader, CSVFormat.DEFAULT.withHeader().withAllowMissingColumnNames());
```

Funkcje dotyczące nagłówków

- dodanie nagłówków

```
withHeader("Col 1", "Col 2")
```

- pierwszy wiersz jako nagłówek

```
withFirstRecordAsHeader()
```

- ignorowanie wielkość liter w nagłówkach

```
withIgnoreHeaderCase()
```

Funkcje dotyczące formatu

- ustawienie niestandardowego separatora

```
withDelimiter(char delimiter)
```

- usuwanie białych znaków

```
withTrim()
```

- określenie cytowanych wartości

```
withQuote(char quoteChar)
```

- definiowanie znaku escape dla cytatów

```
withEscape(char escapeChar)
```

ALTERNATYWY

OPENCSV

https://opencsv.sourceforge.net
Aktualna wersja: 5.9 | 2023-11-21

CSVJDBC

https://github.com/simoc/csvjdbc
Ostatnia aktualizacja: 2024-09-27

OSTERMILLER CSV

https://ostermiller.org/utils/CSV.html
Aktualna wersja: 1.07.00

PODSUMOWANIE

Apache Commons CSV to wydajne i funkcjonalne rozwiązanie dla każdego, kto pracuje z plikami CSV. Biblioteka ta jest łatwa w użyciu i elastyczna, co pozwala na dostosowanie jej do różnorodnych potrzeb. Jest dobrym wyborem dla programistów poszukujących niezawodnego i intuicyjnego narzędzia do zarządzania danymi CSV w środowisku Java.

BIBLIOGRAFIA

- HTTPS://WWW.CALLICODER.COM/JAVA-READ-WRITE-CSV-FILE-APACHE-COMMONS-CSV/
- HTTPS://COMMONS.APACHE.ORG/PROPER/COMMONS-CSV/USER-GUIDE.HTML
- HTTPS://COMMONS.APACHE.ORG/PROPER/COMMONS-CSV/APIDOCS/ORG/APACHE/COMMONS/CSV/CSVFORMAT.HTML#RFC4180
- PRZYKŁADOWY ARKUSZ EXCEL (PLIK KIEROWCY) HTTPS://ARKUSZE.PL/MATURA-INFORMATYKA-2024-MAJ-POZIOM-ROZSZERZONY/

DZIĘKUJEMY ZA UWAGĘ